

Decodability Attack against the Fuzzy Commitment Scheme with Public Feature Transforms

Benjamin Tams*

September 23, 2014

Abstract

The *fuzzy commitment scheme* is a cryptographic primitive that can be used to store biometric templates being encoded as fixed-length feature vectors protected. If multiple *related* records generated from the same biometric instance can be intercepted, their correspondence can be determined using the *decodability attack*. In 2011, Kelkboom *et al.* proposed to pass the feature vectors through a record-specific but public permutation process in order to prevent this attack. In this paper, it is shown that this countermeasure enables another attack also analyzed by Simoens *et al.* in 2009 which can even ease an adversary to fully break two related records. The attack may only be feasible if the protected feature vectors have a reasonably small Hamming distance; yet, implementations and security analyses must account for this risk. This paper furthermore discusses that by means of a public transformation, the attack cannot be prevented in a binary fuzzy commitment scheme based on linear codes. Fortunately, such transformations can be generated for the non-binary case. In order to still be able to protect binary feature vectors, one may consider to use the *improved fuzzy vault scheme* by Dodis *et al.* which may be secured against linkability attacks using observations made by Merkle and Tams.

Keywords

fuzzy commitment scheme, cross-matching, countermeasures

1 Introduction

In 1999, Juels and Wattenberg [1] proposed the *fuzzy commitment scheme* which is considered as an alternative to cryptographic password hashes in

*B. Tams is with the Institute for Mathematical Stochastics, University of Goettingen, Goldschmidtstr. 7, D-37077, Goettingen, Germany. Phone: +49-(0)551-3913515. Email: btams@math.uni-goettingen.de.

which passwords are replaced by samples measured from an individual's biometric characteristic such as, for example, his eyes' *irises* [2]. Given a biometric template the fuzzy commitment scheme can be used to generate a protected record from which it is hard to derive the template unless a sufficiently similar template is presented. Making certain assumptions on the distribution of the biometric templates, the fuzzy commitment scheme provably provides a certain resistance against *irreversibility attacks*. There are, however, more risks than mere irreversibility attack scenarios. For example, the ISO/IEC 24745:2011 international standard [3] explicitly prohibits that the relation between records generated from the same biometric characteristic can be recognized, *i.e.*, the *unlinability requirement*. In fact, as for example investigated by Simoens *et al.* [4], an intruder can link two (or more) related records generated by a fuzzy commitment scheme via the *decodability attack*. In order to prevent the decodability attack, Kelkboom *et al.* [5] proposed to pass the biometric templates through a record-specific but public permutation process.

On the other hand, the countermeasure makes the scheme vulnerable to a *generalized decodability attack* also considered in [4]. More precisely, even though the performance of decodability attack-based cross-matching is effectively reduced, an adversary may still be able to recognize records protecting very similar templates and may now even recover the biometric templates explicitly. To the best of the author's knowledge, this observation is novel.

1.1 Contribution and Outline

After having briefly discussed work that is related to the topic of this paper, in Section 2 we review the functioning of the fuzzy commitment scheme, repeat the decodability attack, the countermeasure proposed by Kelkboom *et al.* [5], and the generalized decodability attack. In Section 3 we show how the countermeasure can be exploited by an intruder to run the generalized decodability attack and demonstrate its applicability by reporting experimental results. Furthermore, we discuss problems with designing effective and feasible countermeasures in a binary fuzzy commitment scheme. In Section 4 we propose a modification of the countermeasure proposed in [5] that effectively prevents a general class of decodability attacks against non-binary fuzzy commitment schemes. For the binary case, we briefly discuss some simple approaches to fix the problem and finally propose to consider the possibility of using the *improved fuzzy vault scheme* by Dodis *et al.* [6] for template protection which, in combination with ideas given by Merkle and Tams [7], can be secured against known linkability attacks. Conclusions and an outlook are given in Section 5. The appendix contains an example of reproducing our experiments using our C++ software library that has been made public for download.

1.2 Related Work

There are other schemes, for example the *fuzzy vault scheme* [8,9] that, as the fuzzy commitment scheme, belongs to a general class of protection schemes called *fuzzy sketches* [6]. As for the fuzzy commitment scheme, there do also exist attacks via record multiplicity against these schemes [7,10,11]. Blanton and Aliasgari worked out the fact that none of these can be safely reused unless being encrypted with secret keys [12]. In fact, using information-theoretic arguments one may argue that, by means of public transformations, the problem of record multiplicity attacks cannot be solved. On the other hand, we stress that these measures still have the potential to render the computational complexity of such attacks infeasible.

2 Preliminaries

2.1 Notation

Throughout, by \mathbf{F} we denote a fixed finite field. By \mathbf{F}^n we denote the set of all column vectors over \mathbf{F} of length n ; for $v \in \mathbf{F}^n$ we denote by v^\top the corresponding transposed row vector and by $|v|$ we denote the *Hamming weight* of v , *i.e.*, the number of positions in v that are non-zero. By $\mathbf{F}^{n \times k}$ we denote the set of all matrices over \mathbf{F} with n rows and k columns. Given a linear error-correcting code $\mathbf{C} \subset \mathbf{F}^n$, we denote by $G \in \mathbf{F}^{n \times k}$ one of its generator matrices such that $\mathbf{C} = \{G \cdot m \mid m \in \mathbf{F}^k\}$ (note that in the literature there exists also the convention to define the transposed G^\top as the generator matrix); consequently, we denote a check matrix of \mathbf{C} by a matrix $H \in \mathbf{F}^{(n-k) \times n}$ of full rank such that $H \cdot G = 0$.

2.2 The Fuzzy Commitment Scheme

In this paper, we consider fuzzy commitment schemes based on linear error-correcting codes over an arbitrary finite field \mathbf{F} . Let $\mathbf{C} \subset \mathbf{F}^n$ be an (n, k, d) -error-correcting code of block length n , dimension k and minimal distance d such that it is capable of rounding every $r \in \mathbf{F}^n$ to its closest codeword c provided that $|c - r| \leq (d - 1)/2$. For more details on the concept of error-correcting codes we refer to one among the good textbooks available in the literature, for example [13].

Let $w \in \mathbf{F}^n$ be a *feature vector* encoding a biometric template extracted from some biometric modality. A codeword $c \in \mathbf{C}$ is selected randomly and is then used to protect w by publishing the *fuzzy commitment* $\mathbf{f} = c + w$. In order to allow safe recovery of the correct template on genuine verification, we sometimes store a cryptographic hash value $h(c)$ along with the fuzzy commitment such that $(\mathbf{f}, h(c))$ is considered as the protected record.

If we assume, for simplicity, that the biometric feature vectors are distributed uniformly among all $w \in \mathbf{F}^n$. Then finding the correct template

w from the record \mathbf{f} is as hard as iterating through all codewords in \mathbf{C} or, if accessible, reverting the cryptographic hash $h(c)$ [1]. It is important to note, that in an analysis the assumption that feature vectors are distributed uniformly among all \mathbf{F}^n typically leads to severe overestimation of system security.

On verification, an allegedly genuine user provides a template $w' \in \mathbf{F}^n$ that is used to be verified against the record $(c + w, h(c))$. If $|w - w'| \leq (d-1)/2$, then $c + w - w'$ differs from c in not more than $(d-1)/2$ positions, thus can be rounded to c using the decoder; the correctness of the codeword c can be verified using $h(c)$ if part of the record, which is deemed to be an accept. Otherwise, if $|w - w'| > (d-1)/2$, the decoder might output a codeword c' that (most likely) is different from the correct codeword c , or fail to decode; both cases are considered as a reject.

2.3 Decodability Attack

Assume that an intruder has intercepted two fuzzy commitments $\mathbf{f}_1 = c_1 + w_1$ and $\mathbf{f}_2 = c_2 + w_2$ and wants to decide whether they are *related*, *i.e.*, whether they protect templates w_1 and w_2 with $|w_1 - w_2| \leq (d-1)/2$; note that, the adversary does not necessarily aim at recovering w_1 or w_2 . Therefore, he computes the offset $r = \mathbf{f}_1 - \mathbf{f}_2 = (c_1 - c_2) + (w_1 - w_2)$. Since \mathbf{C} is linear, $c_1 - c_2 \in \mathbf{C}$. Thus, if $|w_1 - w_2| \leq (d-1)/2$, the offset r can be successfully decoded to a valid codeword; otherwise, if $|w_1 - w_2| > (d-1)/2$, the decoding attempt is supposed to output a valid codeword with the non-zero probability $|\mathbf{F}|^{k-n} \sum_{j=0}^{\lfloor (d-1)/2 \rfloor} (|\mathbf{F}| - 1)^j \binom{n}{j}$ called *sphere packing density* of \mathbf{C} . Consequently, if the sphere packing density is small (or even negligible), the adversary can assume that decoding r yields a valid codeword only if \mathbf{f}_1 and \mathbf{f}_2 are related.

Since the sphere packing density is non-zero, the decodability attack might falsely label two non-related records as related. Therefore, the performance of decodability attack-based cross-matching performance is worse than the system's operational performance [4, 5]. On the other hand, if the sphere packing density is negligible, the cross-matching performance can be very close to the operational performance.

2.4 Prevention via Record-Specific Permutation Processes

In 2011, Kelkboom *et al.* [5] proposed the incorporation of a public record-specific permutation process to prevent the decodability attack. Therefore, let $P \in \mathbf{F}^{n \times n}$ be a permutation matrix. Instead of letting $c + w$ be the fuzzy commitment, we set $\mathbf{f} = c + Pw$ and publish (\mathbf{f}, P) as the record.

Now assume that an adversary has intercepted two records (\mathbf{f}_1, P_1) and (\mathbf{f}_2, P_2) where $\mathbf{f}_1 = c_1 + P_1 w_1$ and $\mathbf{f}_2 = c_2 + P_2 w_2$. Then, since P_1 and P_2 are random, we can assume that the vectors $P_1 w_1$ and $P_2 w_2$ are random.

For the offset $r = \mathbf{f}_1 - \mathbf{f}_2$ to be decodable, $|P_1 w_1 - P_2 w_2| \leq (d-1)/2$ must be fulfilled the probability of which can be estimated with the sphere packing density of \mathbf{C} . Thus, the cross-matching performance based on direct application of the decodability attack becomes close to random.

Kelkboom *et al.* [5] have not considered the following generalization of the decodability attack which, in fact, can be applied by an adversary to determine whether two records protect feature vectors of small difference and even recover them explicitly.

2.5 Generalized Decodability Attack

Simoens *et al.* [4] also considered the case in which the adversary has intercepted two records $\mathbf{f}_1 = c_1 + w_1$ and $\mathbf{f}_2 = c_2 + w_2$ where c_1 and c_2 are codewords from different codes \mathbf{C}_1 and \mathbf{C}_2 , respectively. As before let $r = \mathbf{f}_1 - \mathbf{f}_2$. Denote by $G_1 \in \mathbf{F}^{n \times k_1}$ and $G_2 \in \mathbf{F}^{n \times k_2}$ generator matrices for \mathbf{C}_1 and \mathbf{C}_2 , respectively, let $\tilde{G} = (G_1 | G_2) \in \mathbf{F}^{n \times (k_1 + k_2)}$ be the concatenation of the columns of G_1 and G_2 , and write $e = w_1 - w_2$ for the error pattern between the feature vectors. If we write $\tilde{\mathbf{C}} = \{ \tilde{G}m \mid m \in \mathbf{F}^{k_1 + k_2} \}$ for the code being generated by \tilde{G} , then attempting to decode r in $\tilde{\mathbf{C}}$ leads to a *generalized decodability attack*.

In general it is not clear how to efficiently decode the offset r , even if efficient decoders are available for \mathbf{C}_1 and \mathbf{C}_2 . However, if the difference between the feature vectors is small, *i.e.*, if the error pattern $e = w_1 - w_2$ is of small Hamming weight, then *maximum likelihood decoding* may be a feasible solution to implement the decodability attack. In the following, we describe the procedure in more detail.

Iterate through all $e_1, e_2, \dots \in \mathbf{F}^k$ of Hamming weight less than or equals a small bound b . If in the j th iteration $e_j = w_1 - w_2$, then $\tilde{c} = r - e_j \in \tilde{\mathbf{C}}$ and we may label \mathbf{f}_1 and \mathbf{f}_2 as related; otherwise, if for all tested e_j we have $r - e_j \notin \tilde{\mathbf{C}}$, the records are labeled as non-related.

Irreversibility

If the correct error pattern $e_j = w_1 - w_2$ can be recovered and if the matrix \tilde{G} has rank $\tilde{k} = k_1 + k_2$, then we can even fully recover the feature vectors w_1 and w_2 . More specifically, we can find a unique $\tilde{m} \in \mathbf{F}^{k_1 + k_2}$ solving $\tilde{c} = \tilde{G}\tilde{m}$. Write $\tilde{m}^\top = (m_1^\top \mid m_2^\top)$ where $m_1 \in \mathbf{F}^{k_1}$ and $m_2 \in \mathbf{F}^{k_2}$. Then, $c_1 = G_1 m_1$ and $c_2 = G_2 m_2$ (note that, if hash values $h(c_1)$ and $h(c_2)$ are stored along with \mathbf{f}_1 and \mathbf{f}_2 , respectively, the correctness of c_1 and c_2 can even be verified). Finally, the feature vectors can be determined via the relations $w_1 = \mathbf{f}_1 - c_1$ and $w_2 = \mathbf{f}_2 - c_2$.

If \tilde{G} has rank $\tilde{k} < k_1 + k_2$, the solution of \tilde{m} is not unique. However, there exists such an \tilde{m} yielding the correct c_1 and c_2 . Therefore, depending on the rank of \tilde{G} , the adversary gains advantage for fully recovering feature

vectors w_1 and w_2 of small Hamming distance from \mathbf{f}_1 and \mathbf{f}_2 .

3 Decodability Attack in a Fuzzy Commitment Scheme with Public Record-Specific Permutation

In this section, we consider a simple transformation from the problem of cross-matching two records generated by the fuzzy commitment scheme of which feature vectors have been passed through public record-specific permutation processes to the problem of cross-matching two records generated by the fuzzy commitment scheme based on different codes.

3.1 The Attack

Let (\mathbf{f}_1, P_1) and (\mathbf{f}_2, P_2) be two records generated with a fuzzy commitment scheme based on an (n, k, d) -code \mathbf{C} over the field \mathbf{F} . More precisely, $\mathbf{f}_1 = c_1 + P_1 w_1$, $\mathbf{f}_2 = c_2 + P_2 w_2$, where $c_1, c_2 \in \mathbf{C}$ and $w_1, w_2 \in \mathbf{F}^n$ both encode biometric templates.

Let $G \in \mathbf{F}^{n \times k}$ be a generator matrix of the code \mathbf{C} and set $G_1 = P_1^{-1}G$ and $G_2 = P_2^{-1}G$. Denote by \mathbf{C}_1 and \mathbf{C}_2 the codes being generated by G_1 and G_2 , respectively. Furthermore, write $c'_1 = P_1^{-1}c_1$ and $c'_2 = P_2^{-1}c_2$ which are codewords in \mathbf{C}_1 and \mathbf{C}_2 , respectively. Then $P_1^{-1}\mathbf{f}_1 = c'_1 + w_1$ and $P_2^{-1}\mathbf{f}_2 = c'_2 + w_2$ are fuzzy commitments constructed over \mathbf{C}_1 and \mathbf{C}_2 , respectively. This situation is equivalent to the attack scenario of Section 2.5 and leads to the following attack.

Algorithm 1 (Modified Decodability Attack).

Input A linear code $\mathbf{C} \subset \mathbf{F}^n$ given by its generator matrix $G \in \mathbf{F}^{n \times k}$; two records (\mathbf{f}_1, P_1) and (\mathbf{f}_2, P_2) where \mathbf{f}_1 and \mathbf{f}_2 are of the form $c_1 + P_1 w_1$ and $c_2 + P_2 w_2$, respectively, where $c_1, c_2 \in \mathbf{C}$, $w_1, w_2 \in \mathbf{F}^n$ and permutation matrices $P_1, P_2 \in \mathbf{F}^{n \times n}$; an integer $b \geq 0$.

Output Either a candidate for (w_1, w_2) or FAILURE.

1. **[Code offset]** Let $r = P_1^{-1}\mathbf{f}_1 - P_2^{-1}\mathbf{f}_2$.
2. **[Check Matrix]** Compute $G_1 = P_1^{-1}G$, $G_2 = P_2^{-1}G$ and set $\tilde{G} = (G_1 | G_2)$; find a matrix of maximal rank such that $\tilde{H} \cdot \tilde{G} = 0$, e.g., by computing a basis for the kernel of \tilde{G}^\top .
3. **[Find error pattern]** Let $e_1, \dots, e_B \in \mathbf{F}^n$ be all vectors of Hamming weight $\leq b$ sorted w.r.t. their Hamming weight, i.e., $|e_1| \leq \dots \leq |e_B|$. Next, successively iterate through the list until a j is found such that $\tilde{H} \cdot (r - e_j) = 0$. If none such j exists, return FAILURE; otherwise continue with the next step.

4. **[Solve linear system]** Compute $\tilde{c} = r - e_j$ and solve $\tilde{G}\tilde{m} = \tilde{c}$ for $\tilde{m} \in \mathbf{F}^{k+k}$. Write $\tilde{m}^\top = (m_1^\top, -m_2^\top)$ where $m_1, m_2 \in \mathbf{F}^k$ and set $c_1^* = P_1 G m_1$ and $c_2^* = P_2 G m_2$.
5. **[Return]** Determine $w_1^* = P_1^{-1} \cdot (\mathbf{f}_1 - c_1^*)$ and $w_2^* = P_2^{-1} \cdot (\mathbf{f}_2 - c_2^*)$ and return (w_1^*, w_2^*) .

For an analysis of the attack, which can be formulated as the attack described in Section 2.5, we refer to Section 5.1 in [4].

3.2 Remarks

Note that Algorithm 1 always returns a feature vector pair if the matrix \tilde{G} has rank n . In this case the outputs may not be very useful. This, however, assumes that $2k \geq n$ which may not be fulfilled for codes that can correct a reasonable number of errors (for example, see the codes listed in Table I in [5]). Also note that, if a cryptographic hash value is stored along with a fuzzy commitment, the attack can easily be modified to only output the correct feature vectors or FAILURE.

3.3 Experiments

To show that Algorithm 1 can successfully attack two related records generated with a fuzzy commitment scheme we conducted some experiments with an own C++ implementation which has been made public for download; see the appendix (Section 6) for more details. Therein, we considered binary BCH codes [13] of block length $n = 31, 63, 127, 255$. The codes were designed such that they had a minimal distance being capable of correcting 10% of bit errors. Furthermore, we considered Hamming weight bounds b as input to the algorithm varying between 0 and 5.

For each tested (\mathbf{C}, b) we distinguished two cases: In the first case, we selected feature vector pairs uniformly at random having a Hamming distance less than or equals b ; in the second case, we selected feature vector pairs uniformly at random without accounting for the Hamming distance. To the first case we refer as the *related case* and the second as the *non-related case*. For each feature vector pair, we selected two bit permutations and two codewords uniformly and independently at random. The entries of the feature vectors were reordered using the permutations and then xored with the selected codewords to obtain two fuzzy commitments. The code's generator matrix, the fuzzy commitments together with their respective permutation processes, and the Hamming weight bound b were input to our implementation of Algorithm 1. Whenever the algorithm output candidates for the feature vectors, we accounted the event as a *linkage*. If, in addition, the pair was correct, we accounted the event as a *recovery*. For most tested (\mathbf{C}, b) we simulated 5000 related and 5000 non-related cases; for $n = 255$ and $b = 5$

Table 1: Performance of the attack (Algorithm 1) determined experimentally on a 1.8 GHz server with sufficient memory

(n, k, d)	b	rel./non-rel. linkage rate	rel./non-rel. rec. rate	rel./non-rel. attack time
(31, 11, 11)	0	100% / 0.06%	51.2% / 0%	0ms / 0ms
	1	100% / 2.94%	49.3% / 0%	0ms / 0ms
	2	100% / 39.1%	45.7% / 0%	0ms / 0ms
	3	100% / 99.2%	25.2% / 0%	0ms / 0ms
	4	100% / 100%	6.36% / 0%	0ms / 0ms
	5	100% / 100%	0.94% / 0%	0ms / 0ms
(63, 24, 15)	0	100% / 0%	50.9% / 0%	0ms / 0ms
	1	100% / 0.06%	49.5% / 0%	0ms / 0ms
	2	100% / 2.82%	49.9% / 0%	1ms / 1ms
	3	100% / 46.1%	43.1% / 0%	3ms / 8ms
	4	100% / 99.9%	20.3% / 0%	9ms / 14ms
	5	100% / 100%	3.26% / 0%	14ms / 14ms
(127, 50, 27)	0	100% / 0%	50.2% / 0%	4ms / 4ms
	1	100% / 0%	49.8% / 0%	4ms / 4ms
	2	100% / 0%	50.8% / 0%	5ms / 7ms
	3	100% / 0.08%	50.0% / 0%	35ms / 137ms
	4	100% / 4.18%	50.5% / 0%	932ms / 3.7s
	5	100% / 62.5%	40.9% / 0%	16s / 54s
(255, 87, 53)	0	100% / 0%	50.3% / 0%	29ms / 28ms
	1	100% / 0%	50.9% / 0%	29ms / 28ms
	2	100% / 0%	50.2% / 0%	35ms / 54ms
	3	100% / 0%	50.6% / 0%	465ms / 1.9s
	4	100% / 0%	49.9% / 0%	25s / 106s
	5	100% / 0%	48.3% / 0%	21min / 81min

only 350 related and 350 non-related cases have been simulated. Thereby, we measured the *average linkage rate*, *average recovery rate* and the *average attack time* for the related and non-related case. The results can be found in Table 1.

Our tests clearly indicate that the fuzzy commitment scheme remains vulnerable to record multiplicity attacks even if the protected feature vectors have been passed through record-specific public permutation processes. For example, for $(n, k) = (127, 50)$, according to our tests it is possible to quite reliably distinguish records that protect feature vectors differing in at most $b = 3$ bit positions from those protecting random feature vectors: In all related cases the attack output a pair, but only in approximately 0.08% of the non-related cases. The time that an adversary has to spent before he

can dismiss a non-related record pair was measured as $137ms$. Furthermore, in nearly 50% of the related cases the attack returned the correct feature vectors: This is due to the fact that the linear systems solved in Step 4 of Algorithm 1 had in average two solutions (*i.e.*, the rank of \tilde{G} was $2k - 1$) of which only one was correct. In fact, in all cases for $n = 63, 127, 255$ observed the linear systems had exactly two solutions (in some cases for $n = 31$ the systems had four solutions). This explains why we measured an approximate maximum of 50% related recovery rate. We stress that, if a cryptographic hash value is accessible along with a fuzzy commitment, then the correct of the solution can be selected in Step 4 and then a related recovery rate of up to 100% can be achieved.

It is important to note the following. Kelkboom *et al.* [5] considered BCH codes that are able to correct $\approx 25\%$ of errors while we consider codes of $\approx 10\%$ error-correcting capability. We stress that the higher the error-correcting capability of a linear code, the lower is its dimension and thus its sphere packing density; consequently, the attack will even be more effective against the codes considered in [5]. This is in accordance with the results one achieves using the C++ code example given in the appendix (Section 6).

3.4 The Binary Case

Our observations and experiments lead to the question whether it is possible to pass the feature vectors through another class of transformation than processes reordering the position of the feature vectors in order to prevent the (generalized) decodability attack. Such class of transforms should preserve the Hamming distance between two feature vectors in order to not affect the verification performance of the system. More specifically, we say that a map $T : \mathbf{F}^n \rightarrow \mathbf{F}^n$ *preserves the Hamming distance* if for all $v_1, v_2 \in \mathbf{F}$ the equality $|T(v_1) - T(v_2)| = |v_1 - v_2|$ holds. In Section 4 we construct such transforms and show that they effectively prevent decodability attacks provided $|\mathbf{F}|$ is of sufficient size. On the other hand, the use of these transforms may be ineffective for very small finite fields, in particular, if $|\mathbf{F}| = 2$. The binary case, however, is a very important one and is therefore considered in this section. Unfortunately, the result of our analysis is that *decodability attacks cannot be prevented in the binary case* using a public and record-specific transformation preserving the Hamming distance because every such transformation essentially is a bit permutation process. More precisely:

Theorem 1. *Let $T : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a map that preserves the Hamming distance. Then there exists a permutation matrix $P \in \{0, 1\}^{n \times n}$ such that $T(v) = Pv \oplus T(0)$ for every $v \in \{0, 1\}^n$.*

Here we denote by $\{0, 1\}$ the field with two elements and by \oplus the addition of vectors with coefficients in $\{0, 1\}$ which can be interpreted as an

exclusive or-operation (note that subtraction is the same as addition over $\{0, 1\}$).

By Theorem 1, every transformation through which feature vectors can be passed are permutations plus a constant shifting vector. Due to the publicity of the transformations an adversary can subtract the constant shift from any fuzzy commitment that he has intercepted. Then Algorithm 1 can be applied to attack related fuzzy commitments as before.

Proof of Theorem 1

First, note that T must be bijective (as a map preserving the Hamming distance) since, otherwise, there would exist distinct vectors $v_1, v_2 \in \{0, 1\}^n$ with $0 = |T(v_1) \oplus T(v_2)| = |v_1 \oplus v_2| \neq 0$ which is a contradiction.

Let $\hat{T} : \{0, 1\}^n \rightarrow \{0, 1\}^n$, $v \mapsto T(v) \oplus T(0)$. Note that $|v \oplus v'| = |T(v) \oplus T(v')| = |T(v) \oplus T(v') \oplus T(0) \oplus T(0)| = |\hat{T}(v) \oplus \hat{T}(v')|$; consequently, \hat{T} preserves the Hamming distance, too, and is thus bijective as well. Furthermore, since $|v| = |v \oplus 0| = |T(v) \oplus T(0)| = |\hat{T}(v)|$, \hat{T} also preserves the Hamming weight.

It remains to show that the map \hat{T} is a bit permutation process to which the following lemma is a key.

Lemma 1. *Let $\hat{T} : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a map that preserves the Hamming distance and the Hamming weight. For every list of distinct unity vectors $e_1, \dots, e_\ell \in \{0, 1\}^n$, $0 \leq \ell \leq n$, the equality $\hat{T}(e_1 \oplus \dots \oplus e_\ell) = \hat{T}(e_1) \oplus \dots \oplus \hat{T}(e_\ell)$ holds.*

Proof. From the preservation of the Hamming weight it follows that $\hat{T}(0) = 0$, so the statement holds for $\ell = 0$. Furthermore, the mappings $\hat{T}(e_1), \dots, \hat{T}(e_\ell)$ are unity vectors and pairwise distinct (since \hat{T} preserves the Hamming weight and is bijective). Note that $|\hat{T}(e_1 \oplus \dots \oplus e_\ell) \oplus \hat{T}(e_j)| = \ell - 1$ and since the $\hat{T}(e_j)$ are all distinct, we obtain

$$\begin{aligned} \ell &> |\hat{T}(e_1 \oplus \dots \oplus e_\ell) \oplus \hat{T}(e_1)| \\ &> |\hat{T}(e_1 \oplus \dots \oplus e_\ell) \oplus \hat{T}(e_1) \oplus \hat{T}(e_2)| \\ &\vdots \\ &> |\hat{T}(e_1 \oplus \dots \oplus e_\ell) \oplus \hat{T}(e_1) \oplus \dots \oplus \hat{T}(e_\ell)| = 0 \end{aligned}$$

and the statement of the lemma follows. \square

Due to the fact that every vector in $\{0, 1\}^n$ can be written as the sum of unity vectors and applying Lemma 1 recursively, it follows that \hat{T} is linear and since \hat{T} maps unity vectors to unity vectors, it must be a bit permutation process. \square

4 Countermeasure

In this section, we show which class of transformations can be used such that decodability attacks can be avoided. The modification may be useful in a fuzzy commitment scheme working with sufficiently large fields ($|\mathbf{F}| \geq 23$, say), but is useless for the binary case.

4.1 Record-Specific Field Permutation Process

Let $w \in \mathbf{F}^n$ encode a template which we want to protect via the fuzzy commitment scheme. As usual, let $c \in \mathbf{C}$ be a codeword of a linear (n, k) -code $\mathbf{C} \subset \mathbf{F}^n$. Prior to the generation of a fuzzy commitment, instead of passing w through a random process that reorders the positions of the vector w , we propose to pass the entries of w through a random bijection $\sigma : \mathbf{F} \rightarrow \mathbf{F}$. More precisely, write $w = (u_1, \dots, u_n)^\top$ with $u_j \in \mathbf{F}$; then we may choose $T : \mathbf{F}^n \rightarrow \mathbf{F}^n$, $(u_1, \dots, u_n)^\top \mapsto (\sigma(u_1), \dots, \sigma(u_n))^\top$ as the transformation. T obviously preserves the Hamming distance between feature vectors and can be used instead of a process reordering the feature vectors' positions as proposed in [5].

4.2 Linear Decodability Attacks

To analyze the effectiveness of using a record-specific transformation as above, we introduce a class of attacks that contain the (generalized) decodability attack as special cases. Let $\mathbf{f}_1 = c_1 + T_1(w_1)$ and $\mathbf{f}_2 = c_2 + T_2(w_2)$ be two records generated by the fuzzy commitment scheme where the feature vectors have been passed through public record-specific transformations T_1 and $T_2 : \mathbf{F}^n \rightarrow \mathbf{F}^n$, respectively. We define a *linear decodability attack* to be an attack that exploits two invertible $Q, R \in \mathbf{F}^{n \times n}$ (that may depend on T_1 and T_2) such that the Hamming distance between any feature vectors w_1 and w_2 is preserved as follows: $|w_1 - w_2| = |Q \cdot T_1(w_1) - R \cdot T_2(w_2)|$. Then from decodability of the offset $Q \cdot \mathbf{f}_1 - R \cdot \mathbf{f}_2$ in the code generated by the matrix $\tilde{G} = (QG|RG)$ related \mathbf{f}_1 and \mathbf{f}_2 can be recognized and even be broken with the generalized decodability attack described in Section 2.5.

For the decodability attack (Section 2.3) the transforms T_1 and T_2 and the matrices Q and R are identities. For the decodability attack against the fuzzy commitment scheme with public record-specific permutation process (Algorithm 1), the transforms T_1 and T_2 are the linear maps induced by the permutation matrices P_1 and P_2 , respectively, while $Q = P_1^{-1}$ and $R = P_2^{-1}$.

4.3 Analysis of the Countermeasure

We next show that the probability for an effective linear decodability attack to exist can be very small if the entries of the feature vectors have been passed through a public, random and record-specific field permutation.

Therefore, note that, in order to ease notation, it is sufficient to assume that only one (the second, say) record protects a feature vector whose entries have been transformed. The key of our estimation is given by the following.

Lemma 2. *Let $T : \mathbf{F} \rightarrow \mathbf{F}$ be a map. Assume that there exists invertible matrices $Q, R \in \mathbf{F}^n$ with $|w_1 - w_2| = |Q \cdot w_1 - R \cdot T(w_2)|$ for every $w_1, w_2 \in \mathbf{F}^n$. Then T is linear.*

Proof. The condition $|w_1 - w_2| = |Q \cdot w_1 - R \cdot T(w_2)|$ for every $w_1, w_2 \in \mathbf{F}^n$ assumes that $0 = Q \cdot w - R \cdot T(w)$ for all $w \in \mathbf{F}^n$. Hence $T(w) = R^{-1} \cdot Q \cdot w$ and the statement of the lemma follows. \square

For a bijective map $\sigma : \mathbf{F} \rightarrow \mathbf{F}$ set $T : \mathbf{F}^n \rightarrow \mathbf{F}^n$, $(u_1, \dots, u_n)^\top \mapsto (\sigma(u_1), \dots, \sigma(u_n))^\top$. Then, if $|w_1 - w_2| = |Q \cdot w_1 - R \cdot T(w_2)|$ it follows from the lemma that T must be linear and thus its components, *i.e.*, the map $\sigma : \mathbf{F} \rightarrow \mathbf{F}$, must be linear, too, and, as a bijection, additionally be invertible.

There are $(|\mathbf{F}| - 1) \cdot |\mathbf{F}|$ invertible linear maps and $|\mathbf{F}|!$ permutations. Consequently, we estimate the probability that the incorporation of two random field permutation processes enables a linear decodability attack as $1/(|\mathbf{F}| - 2)!$.

For example, for $|\mathbf{F}| = 32, 64$ and 128 , these probabilities evaluate to approximately 2^{-108} , 2^{-284} and 2^{-702} , respectively, which may be reasonably small to prevent an attacker from successfully running a linear decodability attack.

Disclaimer

At this point we stress that, although linear decodability attacks (as considered in this section) can be avoided by incorporating a public field permutation process, there might exist other effective record multiplicity attacks. Yet, to the best of the author's knowledge, there is no known approach being more efficient than breaking the records individually—even in the case of equal feature vectors. However, future research is needed to confirm or disprove the validity of this assumption.

4.4 The Binary Case

The countermeasure proposed in Section 4.1 is useful only if we can assume that the fields have a sufficient size. However, many applications work with binary feature vectors. On the other hand, we have discussed that preventing decodability attacks via a public transformation preserving the Hamming distance can be problematic in a binary fuzzy commitment scheme.

One solution to the problem would be to use non-linear error-correcting codes. Then the decodability attack cannot be applied which exploits linearity of the underlying error-correcting code. However, the countermeasure

proposed by Kelkboom *et al.* [5] was meant to address the most dominant linear case.

In the binary and linear case, in combination with record-specific but public permutation processes, it is possible to flip a few bits randomly such that the error pattern that an adversary needs to correctly guess during the attack is expected to have at least a certain Hamming weight. In such a way the attack might be rendered infeasible. On the other hand, the flipped bits introduce random errors that need to be compensated by a higher error-correction capability of the code and thus a lower dimension which negatively affects irreversibility of the records. Yet, if the dimension is still sufficient to guarantee a certain security, randomly flipping a few bits may be a valid countermeasure for the binary case. Nonetheless, other promising solutions do exist.

Switch the Scheme

Another way to achieve resistance against linkability attacks for binary feature vectors may be to map them to *feature sets*, *i.e.*, subsets of a finite field, which can be protected using a *fuzzy vault scheme* [8]. Therein, each position of the feature vector is attached with a unique field element and for each position in the feature vector that is set to one the feature set is defined to contain the field element attached to the bit position; this relation has already been outlined in [6]. The feature set can then be protected using the *improved fuzzy vault scheme* by Dodis *et al.* [6]. It is important to note that there exist record multiplicity attacks against the improved fuzzy vault scheme as, too [7]; in fact, each known biometric template protection scheme is in principle vulnerable to linkability attacks [12]. However, since the improved fuzzy vault scheme may work with fields containing more than two elements it is possible to pass the feature elements through a record-specific field permutation process [7] similar to the measure briefly discussed above (Section 4.1) in a non-binary fuzzy commitment scheme. It is important to note that when switching to the improved fuzzy vault scheme, fulfilling the unlinkability requirement may not be at the cost of verification performance or security and is therefore, in our view, a very promising fix to the problem. Yet, a proof that passing feature elements through a record-specific (but public) field permutation can effectively prevent any linkability attack would be desirable, *e.g.*, by proving a complexity-theoretical statement such as “the publicity of the permutation processes cannot be exploited to gain advantage in linking two records as compared when breaking one the records individually”. Alternatively, we could try to disprove the conjecture by finding an effective linkability attack as a counterexample.

5 Discussion

5.1 Summary

We discussed the effect of preventing decodability attack-based cross-matching in a fuzzy commitment scheme by incorporating a random public bit permutation process [5]. We found that the measure does not completely solve the problem and even can make the scheme vulnerable to reversibility attacks, if the protected feature vectors have a sufficiently small Hamming distance. We supported this observation experimentally and complemented our work by proving that there exists no class of transformations preserving the Hamming distance between feature vectors that solve the problem of decodability attacks in a binary fuzzy commitment scheme. On the other hand, we were able to construct effective transformations provided the underlying finite field is of a sufficient size. For the important binary case, we gave arguments that the improved fuzzy vault scheme by Dodis *et al.* [6], in combination with ideas established by Merkle and Tams [7], is a possible alternative to protect binary feature vectors in an unlinkable and key-less biometric template protection system.

5.2 Conclusion

Our work clearly states that the mere incorporation of a record-specific bit permutation process is not sufficient in order to make a binary fuzzy commitment scheme resistant against linkability attacks unless the ineffectiveness of Algorithm 1 can be guaranteed in the security analysis of a specific implementation: For example, by estimating the probability as negligible that related binary biometric feature vectors agree in less than a certain number of bits (depending on the aimed security level) or by guaranteeing that the code $\tilde{\mathbf{C}}$ in the algorithm has a sufficiently large sphere packing density. If such assertions cannot be made, the protection of binary feature vectors with a fuzzy commitment scheme seems not to be possible if based on linear error-correcting codes. Even though we discussed some possible approaches in order to prevent decodability attacks, future research is needed to confirm whether they yield resistance against heavy attacks.

5.3 Outlook

In particular, we briefly discussed the use of the improved fuzzy vault scheme by Dodis *et al.* [6] as a possible alternative to implement a key-less and unlinkable protection for binary feature vectors: Even though there exists an effective attack in the presence of record multiplicity against the improved fuzzy vault scheme, there also exists a possible countermeasure [7] similar to the one discussed in Section 4.1. As part of our future research, we plan to prove (or disprove) the infeasibility of attacking the improved fuzzy vault

scheme (with countermeasure) from record multiplicity using complexity-theoretical arguments.

ACKNOWLEDGEMENT

The support of the Felix Bernstein Institute for Mathematical Statistics in the Biosciences and the Volkswagen Foundation is gratefully acknowledged.

References

- [1] A. Juels and M. Wattenberg, “A fuzzy commitment scheme,” in *Proc. of ACM Conf. on Computer and Communications Security*, 1999, pp. 28–36.
- [2] F. Hao, R. Anderson, and J. Daugman, “Combining crypto with biometrics effectively,” *IEEE Trans. Comput.*, vol. 55, no. 9, pp. 1081–1088, Sep. 2006.
- [3] ISO/IEC JTC1 SC2 Security Techniques, “ISO/IEC 24745:2011. Information Technology - Security Techniques - Biometric Information Protection,” International Organization for Standardization, 2011.
- [4] K. Simoons, P. Tuyls, and B. Preneel, “Privacy weaknesses in biometric sketches.” in *IEEE Symp. on Security and Privacy*. IEEE Computer Society, 2009, pp. 188–203.
- [5] E. J. C. Kelkboom, J. Breebaart, T. A. M. Kevenaar, I. Buhan, and R. N. Veldhuis, “Preventing the decodability attack based cross-matching in a fuzzy commitment scheme,” *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 1, pp. 107–121, 2011.
- [6] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, “Fuzzy extractors: How to generate strong keys from biometrics and other noisy data,” *SIAM J. Comput.*, vol. 38, no. 1, pp. 97–139, 2008.
- [7] J. Merkle and B. Tams, “Security of the improved fuzzy vault scheme in the presence of record multiplicity (full version),” *CoRR*, vol. abs/1312.5225, 2013, submitted.
- [8] Juels and Sudan, “A fuzzy vault scheme,” in *Proc. Int. Symp. Inf. Theory*, A. Lapidoth and E. Teletar, Eds., 2002, p. 408.
- [9] A. Juels and M. Sudan, “A fuzzy vault scheme,” *Des. Codes Cryptography*, vol. 38, no. 2, pp. 237–257, 2006.
- [10] W. J. Scheirer and T. E. Boulton, “Cracking fuzzy vaults and biometric encryption,” in *Proc. of Biometrics Symp.*, 2007, pp. 1–6.

- [11] A. Kholmatov and B. Yanikoglu, “Realization of correlation attack against the fuzzy vault scheme,” in *Proc. SPIE*, vol. 6819, 2008.
- [12] M. Blanton and M. Aliasgari, “Analysis of reusability of secure sketches and fuzzy extractors,” *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 9, pp. 1433–1445, 2013.
- [13] E. R. Berlekamp, *Algebraic coding theory*. Laguna Hills, CA, USA: Aegean Park Press, 1984.

6 Appendix

We provide a C++ software THIMBLE that can be downloaded from

`http://www.stochastik.math.uni-goettingen.de/biometrics/thimble`

and that is licensed under the LGPL. The library provides a variety of functionalities intended to be useful for research purposes related to biometric template protection. In particular, it provides the `BCHCode` class of which objects represent binary BCH codes which can correct binary vectors, represented by objects from the `BinaryVector` class, to their closest code-words within the BCH code’s error-correcting radius. In this appendix we demonstrate how the experiments in Section 3.3 can be reproduced using THIMBLE and its `BCHCode` class.

An executable program that uses THIMBLE may be of the following form

```
#include <thimble/all.h>

using namespace std;
using namespace thimble;

int main( int argc , char *args[] ) {

    // IMPLEMENT ME

    return 0;
}
```

where `// IMPLEMENT ME` should be replaced by C++ code. To generate an $(n, k, d)=(127, 36, 31)$ -BCH code (which is one of the BCH codes considered in Kelkboom *et al.* [5] being of error-correcting capability $\approx 25\%$), we may run

```
BCHCode C(127,15);

int n = C.getBlockLength();
```



```

int k = C.getDimension();
int d = C.getMinimalDistance();

```

which creates a BCH code of block length 127 being capable of correcting (at least) 15 bit errors and is thus of minimal distance $d = 31$. We may choose two feature vectors of Hamming distance 4, say, at random by

```

BinaryVector w1(n) , w2(n) , e(n);
int hw = 4;

w1.random(true); // first feature vector
e.wrandom(hw,true); // error pattern
w2 = w1 + e; // second feature vector

```

where the arguments **true** advise the `random()` methods to use a strong random generator (*e.g.*, `/dev/urandom` on UNIX-based systems); note that the above C++ code can easily be adjusted to simulate the attack in the non-related case by also choosing the second feature set uniformly at random. Two codewords may be randomly drawn from the BCH code by

```

BinaryVector c1 , c2;

C.random(c1,true);
C.random(c2,true);

```

To generate two random permutation processes, represented by objects from the `Permutation` class, run

```

Permutation P1(n) , P2(n);

P1.random(true);
P2.random(true);

```

Next, we can construct two fuzzy commitments.

```

BinaryVector f1 , f2;

f1 = c1 + P1*w1;
f2 = c2 + P2*w2;

```

Now, assume that we are an adversary who has intercepted the records $(f1, P1)$ and $(f2, P2)$; we keep track of correct feature vectors and codewords to be later able to verify the correctness of an output. With the notation of Algorithm 1, we may obtain generator matrices encoded as `BinaryMatrix` objects for the codes C_1 and C_2 through

```

BinaryMatrix G1 , G2;

G1 = inv(P1) * C.getGeneratorMatrix();
G2 = inv(P2) * C.getGeneratorMatrix();

```

and we can compute its concatenation \tilde{G}

```
BinaryMatrix tG = concatCols(G1,G2);
```

A check matrix \tilde{H} for the code generated by \tilde{G} may be computed through

```
BinaryMatrix tH = transpose(kernel(transpose(tG)));
```

Starting from a zero error pattern e encoded by a **BinaryVector** we may successively iterate through an error pattern sequence of increasing Hamming weight until for the offset $r = P_1^{-1}\mathbf{f}_1 - P_2^{-1}\mathbf{f}_2$ the equality $\tilde{H} \cdot (r - e) = 0$ holds.

```
BinaryVector r = inv(P1) * f1 - inv(P2) * f2;
```

```
// we use the error pattern from above for  
// the guesses of the error pattern  
e.setZero();
```

```
// Keeps track of whether a working error  
// pattern is found.  
bool success = false;
```

```
// Loop as long as the Hamming weight is smaller  
// than 'hw'  
while (e.hammingWeight()<=hw) {
```

```
    if ( isInKernel(r-e,tH) ) {  
        success = true;  
        break;  
    }
```

```
    // next element in the sequence of error  
// patterns of increasing Hamming weight  
    e.next();  
}
```

```
// If no working error pattern has been found,  
// output a message, resembling that the records  
// definitely do not protect feature sets of  
// Hamming distance smaller than or equals 'hw'  
if ( !success ) {  
    cout << "NON-RELATED" << endl;  
    return 0;  
}
```

After the iteration, we may try to find the feature vectors using the relations described in Section 2.5 and Section 3 from the solution of a linear system.

```

BinaryVector m;

// Solve the linear system
solve(m,tG,r-e);

// Message vectors
BinaryVector m1(k) , m2(k);
for ( int i = 0 ; i < k ; i++ ) {
    m1.setAt(i,m.getAt(i));
    m2.setAt(i,m.getAt(i+k));
}

// Codeword candidates
BinaryVector rc1 , rc2;
rc1 = C.getGeneratorMatrix() * m1;
rc2 = C.getGeneratorMatrix() * m2;

// Feature vector candidates
BinaryVector rw1 , rw2;
rw1 = inv(P1)*(f1 - rc1);
rw2 = inv(P2)*(f2 - rc2);

// Check if recovery was successful and
// output a corresponding message.
if ( rw1 == w1 && rw2 == w2 ) {
    cout << "REVERTED" << endl;
} else {
    // If recovery was unsuccessful, then
    // only label the records as related.
    cout << "RELATED" << endl;
}

```

In average, the attack outputs in nearly 50% of the cases **REVERTED** and in the other cases **RELATED**. If the C++ code is adjusted to attack non-related records, then it will with probability very close to 0% output **NON-RELATED**. Note that due to the higher error-correcting capability of $\approx 25\%$ of an (127,36)-BCH code the attack performs better as for the (127,50)-BCH code considered in Table 1 which can correct up to $\approx 10\%$ errors but for which the attack fails in $\approx 4.18\%$. We stress that the C++ code example given in this appendix can be easily adjusted to reproduce our experimental results from Section 3.3.